DOCUMENT RESUME

ED 246 462 CS 208 495

AUTHOR Thiesmeyer, John
TITLE Some Boundary Considerations for Writing-Software.
PUB DATE 14 Apr 84
NOTE 13p.; Paper presented at the University of Minnesota Conference on "Computers and Writing: Research and Applications" (Minneapolis, MN, April 14, 1984).
PUB TYPE Guides - Classroom Use - Guides (For Teachers) (052) -- Speeches/Conference Papers (150) -- Viewpoints (120)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Computer Software; Dictionaries; *Editing; Educational Media; Higher Education; Language Usage; *Microcomputers; Secondary Education; *Spelling; *Word Processing; Writing Instruction
IDENTIFIERS *Software Evaluation; Text Editing

ABSTRACT

Computerized spelling programs or "spelling checkers" can be a wonderful tool for writers at any level of competence. However, they should not be used as adjuncts to the teaching of writing unless they meet two boundary conditions, one of size and one of design. The problem with design of the programs is that they allow for the correction of typographical errors and misspellings without human intervention, thus reducing the student to a passive key-pusher. The problem with size is that most of the spelling dictionaries have a limited working vocabulary. Other writing programs known as "grammar-" or "style-checkers" call attention to incorrect usages, redundancies, wordiness, meaningless intensifiers, gender-specific terms, split compounds, cliches, and other solecisms common in bad writing. Unfortunately, these programs are unregenerately prescriptive, offering substitutions for nearly every phrase they store. Better suited to the needs of a writer would be a software package that analyzes text. However, the high number of computations such a program would require renders such an idea impractical. Writing software packages, if properly designed and applied, can provide extensive text analysis. Unfortunately, much of the software originates with commercial programers rather than with experienced classroom teachers. As such, they may, in fact, produce worse rather than better writers. (HOD)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED246462

(Paper presented at the University of Minnesota Conference on "Computers and Writing: Research and Applications," April 14, 1984)

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
X This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
John E. Thiesmeyer
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

SOME BOUNDARY CONSIDERATIONS FOR WRITING-SOFTWARE

John Thiesmeyer
Dept. of English and Comparative Literature
Hobart and William Smith Colleges
Geneva, NY 14456

In this paper I will briefly discuss a number of present and future limitations on the design of software to be used in conjunction with word processors to help teach writing. The limitations, or boundaries, I have in mind are of several kinds: linguistic, pedagogical, mathematical, occasionally even pragmatic. Let me make clear that my remarks are intended primarily for a teaching-learning application above the elementary level; in reference to a business, home or lower-school environment they would need occasional qualification.

The decline of the text in our culture produces more and more people who have learned vocabulary through the ear not the eye, and whose phonetic understanding is inadequate to a traditional English orthography. Barring radical reform of the rules, writing teachers face an apparently Sisyphean labor. Repeated marking of spelling errors on student papers is demoralizing, when we know the student is unlikely to profit by our labors. As suggested in a recent N. Y. Times Magazine piece (February 26, 1984) on the writing of very young children, moreover, our red ink may do actual

CS208495

damage to the writer. "Snakes are dispikibel," wrote a first-grader, and his principal pointed out that if the misspelling were immediately jumped on, the youth might choose "bad" the next time to be safe. Surely the point is good at any level of teaching: the more frantically we decry and punish poor spelling the more we promote a monosyllabic and unadventurous prose. Yet spelling ability declines each year among college freshmen.

Help is available in the form of "spelling checkers," a wonderful utility for writers at any level of competence. Except for homonyms and split compounds, they can eradicate typographical errors and misspellings in a document. Because they protect rather than punish, the writer can use an entire working vocabulary rather than choosing a safe but smaller orthographic one. The teacher need never again confront a set of papers averaging three or more misspellings per page; my own experience shows that three or four spelling errors per set is an achievable goal.

Despite these promises, however, I would argue that spelling checkers should not be used as adjuncts to the teaching of writing unless they meet two boundary conditions, one of size and one of design.

Let me take up the design problem first. The writers of spelling-checker programs have figured out that in addition to tagging misspelled words their programs can easily be made to offer "corrections" of one kind or another. If hte is a common typographical error for the, for example, the checker can be programmed either to make the correct substitution silently or to offer it as an option to the user, who need only touch a key to have the replacement made. The logical extension is obviously to incorporate a subsidiary dictionary of common typos and misspellings (a dictionary that could be indefinitely large, in principle) and make or offer corrections for all of them. Another strategy is to

store phonetic approximations along with correct English letterings as an aid to the increasing numbers of aural spellers. A third approach is to search the main dictionary list for words spelled like the word in question, to some degree of approximation, and offer a menu of possibilities for the user to choose among; thus if seperate turns up in a text the menu will probably include the correct separate among its suggestions. All three types of assistance are presently used in commercial spelling programs, and sophisticated algorithms are being developed that will eventually allow perhaps 90% or more of the typographical errors and misspellings in a document to be corrected automatically, without human intervention (see, e. g., Communications of the Association for Computing Machinery [Comm ACM hereafter], Apr. 1984).

The problem with strategies like these is that they are thoroughly anheuristic. Instead of promoting learning they deny initiative and forestall thought: the student (or any user) is reduced to a passive key-pusher, or even an uninvolved bystander. As a teacher, I would place all checker programs promising "correction" beyond the pale, unless their correcting features can be disabled for pedagogical use. If students are forced to look up each rejected word there is hope that over time the conventional orthographies may be learned; with optional or automatic "correction," learning is precluded and the spelling problem in our culture can only worsen.

The question of appropriate size for a spelling dictionary is interesting because there is confusion over just how many words we use. If we accept the common estimate of around 15 thousand words as the typical working vocabulary of a high-school senior (see E. L. Thorndike and I. Lorge, The Teacher's Word Book of 30,000 Words, 1944), it would seem that a

dictionary of 25,000 words or so is ample to catch most misspellings. A computer professionals' journal suggested three years ago that "a dictionary of 10,000 words would be quite reasonable for a small community of users" (Comm ACM, Dec. 1980). Programmers' practice generally concurs. To instance only a few of the fifty or more spelling checkers currently available, Commodore advertises Totl-Speller with 10,000 words; Aspen Software (now Wang Electronic Publications) has 38,000 in its Proofreader; Oasis Systems offers 45,000 with The Word Plus. Among the large computer systems I know of, the spelling dictionaries on VAX/VMS minicomputers and on mainframes using the Unix operating system have typically contained 20,000 words.

A more recent estimate is that the average high-school senior knows about 7000 root words (E. F. Dupuy, The Rationale, Development and Standardization of a Basic Word Vocabulary Test, 1974); the multiplier effect of prefixes, suffixes and compounding would suggest that the average working vocabulary is considerably greater than 15,000. Fifty years ago, indeed, Leonard Bloomfield maintained that even uneducated adult speakers use "somewhere round 20,000 to 30,000" words (Language, 1933).

I cannot refine these estimates for the individual student, but the discvery I (and probably others) have made is that the working vocabulary of a group of 20-odd quite ordinary college students is much closer to 100,000 than to 15,000 words. It follows that all the popular checkers mentioned, along with most others now on the market, are grossly inadequate for teaching purposes. This somewhat surprising assertion is based on my experience with two sections of 20 students each, using a spelling checker (Radio Shack's Scripsit Dictionary) with a barely adequate 75,000 words. That is the largest spelling dictionary I know of for "8-bit" micros, yet

the 400 papers my students wrote averaged about one common vocabulary word apiece that the spelling dictionary did not recognize. The pedagogical problem is that if, as a result of using an inadequate spelling program, students are forced to look up a fair number of correctly spelled words, they will soon become cynical about the process and prone to guess that some of their incorrect spellings are correct and need not be looked up. When that happens, the numbers of misspelled words will slowly creep upward again in their papers.

I should add a final warning to those tempted to buy a spelling checker for personal or classroom use. The often-advertised feature that allows a user to "add as many words as desired" to a spelling checker is a snare. The time involved in adding even two or three thousand correctly spelled words to a checker dictionary, after first suspecting and then checking to confirm they are missing, is prohibitively long, and since as I have indicated most checkers are too small by half or more, making them adequate is a task not even to be contemplated by a busy person.

As writing teachers concerned with the two kinds of limitations I have described for spelling programs--a lower bound on their adequacy at around 80 or 100 thousand words, and a heuristic need to prevent mindless correction features--the scope of the assault we face at programmers' hands is best seen, perhaps, in contemplation of the best-known checkers. Broderbund's Bank Street Speller, devised to accompany the widely praised word-processing program for children, offers corrections; so does Cornucopia's Electric Webster, proudly advertised in the words of a reviewer as "the Cadillac of vocabulary programs"; the legendary Writer's Workbench, which Bell Labs is said to be planning at last to issue in a version suitable for microcomputers after lengthy mainframe development and

use, includes a spelling checker with only 30,000 words; and whoever purchases MicroPro's SpellStar to accompany their popular Wordstar writing program, will get a mere 20,000 words for a list price of $250.

The next level of software that helps with writing is constituted by the so-called "grammar-" or "style-checkers," which are neither. Like spelling checkers they depend on stored dictionaries, in this case primarily of phrases rather than individual words. Whereas spelling dictionaries hold lists of correctly spelled words and call attention only to spellings that don't find a match, the phrase checkers contain incorrect usages (different then), redundancies (time period), wordiness (due to the fact that), meaningless intensifiers (incredible), gender-specific terms (mankind), split compounds (some what), clichés (pure and simple), and other solecisms common in bad writing. Many words and phrases in these dictionaries raise questions of taste or judgment rather than of outright error. When a match is found between text and phrase dictionary the potential mistake is reported to the user for reconsideration.

Usage programs can also check for some mechanical errors, like the placing of a period or comma outside close-quotation marks, and can keep tallies of words or phrases singled out for special attention, such as copulative verbs or referential pronouns. Some will print a concordance of the writer's text, useful for finding examples of excessive repetition.

Misused words and phrases constitute bad grammar only sometimes, and their revision is not guaranteed to elevate style. The usage checkers do have a salutary effect on student writing, however, beyond their efficacy in removing many blighted locutions before the teacher has to respond to a

paper, they help considerably in getting the point across that extensive analysis, revision and rewriting should always precede submission.

I don't propose to dwell at length on usage checkers here. My wife Elaine (an English teacher and computer programmer) and I have been for many months intensively involved in the development and use of such a system for teaching, and I am reporting on it elsewhere at this conference. Commercial examples include Electric Webster's "Grammar Option," Oasis System's Punctuation + Style (with an excellent checker for mechanical errors), and Wang Electronic Publications' Grammatik, all of which I have studied, and one of the components of Writer's Workbench. All include phrase dictionaries of between 600 and 750 items; the one we have been using is about three times as large. (I estimate that the dictionary size needs to be at least doubled again, to 4000-5000 entries, before the usage-checker programs can be reasonably sure of catching most common errors.)

The boundary consideration that I find applies in the evaluation of such software for classroom and laboratory use is similar to one I raised about spelling programs: the usage checkers are unregenerately prescriptive, offering substitutions for nearly every phrase they store. This is bad enough in its denial of the user's creative faculties, its implication that for each misuse there are only one or sometimes two appropriate corrections, its cultivation of a kind of bland, lowest-common-denominator prose.

The problem is compounded when, as in the examples I have seen, the phrase dictionaries were evidently put together in haste, and not by experienced teachers or good writers. Thus in Grammatik the word busboy is identified as a "gender specific [sic] term," which it is, accompanied by

the admonition to "use 'server,'" which is incorrect. Or we find excesses like Electric Webster's message, each time the writer uses the word put, that there is a similar word putt (EW calls it a "homonym," which it is not); so for every golfer who cannot spell the final stroke there are surely dozens, perhaps hundreds, of writers whose perfectly proper put's would call forth specious messages. Again: in all checkers I have examined a particular kind of oversight occurs repeatedly: a verb or verb phrase we might all agree needs revision is listed, and therefore picked up by the program, only in its first-person-present or infinitive form. Thus utilize is flagged, and use is suggested to replace it, but the writer who uses utilizes, utilized, utilizing, utilization or even utilizer is ignored. Such oversights are the rule, not the exception, when programmers usurp the role of writing teacher.

Until software developers begin taking writers' needs more thoughtfully into account, then, the teacher wishing to experiment with a usage checker should make sure its phrases can be extensively edited and supplemented, its prescriptions suppressed. Unlike Electric Webster's "Grammar Option" it should also, by the way, allow its messages to be printed out: a cathode-ray tube with a few lines of text, an error message and a blinking cursor is no proper environment for thoughtful revision of one's work. I may say that Grammatik, which is the much-modified heart of our own system, is quite brilliantly designed to allow such options, though there are serious deficiencies in its analyses of English usage and mechanics.

Beyond the levels of word and phrase checking already incorporated in writing software is the level of what might be called "phrase pattern" at which, for example, a sentence including the words not only would be

checked to make sure it had a following but or but also. Conversely, a sentence containing one, everyone, or person and a subsequent they, them, or their would be gently queried about agreement. So far as I know, text analysis at this level has scarcely begun among academic programmers, much less commercial software developers; I have been accumulating instances of such patterns and we hope to develop a program this summer to incorporate them.

Despite my criticisms and reservations I have high hopes for writing-software. Properly designed and applied, it can provide extensive text analysis to the student under circumstances which encourage revision, and can do so without preempting the student's own initiative or creativity. Although hard evidence of long-term benefits is yet to accumulate, classroom experience so far is encouraging. The hope that some of the pitfalls I have outlined can be avoided simply by leaving writing-software development in academic hands is dashed, however, by the reflection that the probable future of word processing lies not with academic mainframes or minis but with microcomputers, and that much of the writing software students use will therefore originate with commercial programmers. I hope I have adequately communicated my sense that commercially produced computer-assisted instruction in writing may well produce worse rather than better writers.

As a final consideration for this paper it is worth raising the question just how sophisticated we might expect computer text-analysis eventually to become. Will we see full-fledged syntax software unerringly picking out sentence fragments, comma splices, dangling modifiers, improperly formed possessives, failures of agreement between subject and

verb, noun and pronoun? For several years tantalizing reports have come out of IBM research labs at Yorktown Heights, N.Y., of a text processor called "Epistle" that has remarkable powers of syntactic analysis (see "Studies in Text Processing," IBM Research Highlights, Oct. 1981, and Forbes, Aug. 15, 1983). A recent report on Japan's "Fifth-Generation Project" to become world leader in advanced computer technology indicates that by the early 1990's the Japanese hope to have a system with a "vocabulary of up to 10,000 words, 2000 grammar rules, and 99% accuracy in syntactic analysis of written natural language. . . ." The system will employ computers capable of performing up to a billion logical inferences per second, perhaps 30,000 times faster than today's best machines (Comm ACM, Sept. 1983).

In the face of such prospects, who would dare suggest that natural-language analysis approximating that of a human expert will not soon be carried out by machines? I would, for one. I cannot prove the case definitively, but with the help of a travelling-salesman story I owe to my colleague Richard Decker in Mathematics, I will offer a strong conjecture.

It has been shown machematically that a "context-free" language can be "computed," that is, it can in principle be generated and analyzed by a computer (see Stephen A. Cook's Turing Award Lecture, "An Overview of Computational Complexity," Comm ACM, June 1983). No such proof exists for context-bound languages. English is not context-free, as a pair of examples will show. "'How time flies,' sighed Susy's mother as the child, using her birthday stopwatch, learned how to time flies." In the context of the one small word to, the words time and flies exchange their parts of speech. Again: the words does she almost invariably indicate a question,

unless the prior context includes a phrase starting with only: "Only after long deliberation does she invest in hog futures."

Though but two of a myriad examples of the ways in which word arrangements alter meaning in English, these should suffice to indicate how very many special rules and exceptions, in addition to the complicated taxonomies of our simple, compound, complex and compound-complex forms, would be needed for a full syntactic description of the language.

The lexical ambiguities of our language mean, furthermore, that in addition to a very large body of rules, a competent sentence analyzer would need a dictionary that not only listed but labelled each word according to its allowable syntactic functions. A sentence analysis might then involve trying all permitted lexical functions of its words in order to match its structure against the stored rules. All examples I have seen of natural-language parsers use labelled lexicons of this kind (a good example of what is being attempted is Jane Robinson's "Diagram: A Grammar for Dialogues," Comm ACM, Jan. 1982). As we shall see in a moment, the discovery of an apparent lower limit on the size of effective spelling checkers has considerable bearing on the practical possibilities for natural-language analysis.

For a dramatic illustration of what happens when a computer must try many arrangements or combinations of elements to find a desired solution, I offer the travelling-salesman problem. Consider the following "real-world" situation. A company specializing in the manufacture of very large computer systems has sales offices in 40 American cities. The sales manager wishes to visit each office on an annual inspection tour. Fuel prices are high, and time is money. Can the company computers calculate the shortest route among the forty cities?

Intuition suggests that a forty-city route should be a simple matter for a modern mainframe number-cruncher. Mathematics replies that the problem cannot be solved in practice: the number of alternate routes to be compared is greater than 4×10^{47}, and the fastest conceivable computer could not complete the calculation during the estimated life of the universe.

As the travelling-salesman problem illustrates, when the number of items goes up the number of possible combinations of items increases enormously. I know of no one who has worked out the mathematics for word combinations and sentence types, but reflection suggests that the computational requirements will be high. If the working vocabulary of a group of ordinary teenagers is somewhere around 100,000 words, we can see that the 10,000-word lexicon anticipated by the Japanese for their natural-language analyzer may be an order of magnitude too small, even for a limited speech community. If the number of words to be checked and the number of phrase and sentence rules are high enough, I conjecture that the sheer magnitude of the computational task will render the dream of a program that identifies all, or even most, incorrect sentences impractical for a long while to come.